Detection, Diagnosis, and Recovery From Clock-Domain Crossing Failures in Multiclock SoCs

Naghmeh Karimi, *Member, IEEE,* and Krishnendu Chakrabarty, *Fellow, IEEE*

***Abstract*—Clock-domain crossing (CDC) faults require careful post-silicon testing for multiclock circuits. Even when robust design methods based on synchronizers and design verification techniques are used, process variations can introduce subtle timing problems that affect data transfer across clock-domain boundaries for fabricated chips. We integrate solutions for detecting and locating CDC faults, and ensuring post-silicon recovery from CDC failures. In the proposed method, CDC faults are located using a CDC-fault dictionary, and their impact is masked using post-silicon clock-path tuning. To quantify the impact of process variations in the transfer of data at clock domain boundaries of multiclock circuits and to validate the proposed error-recovery method, we conducted a series of HSpice simulations using a 45-nm technology. The results demonstrate high incidence of process variation-induced violation of setup and hold time at the boundary flip-flops, even when synchronizer flip-flops are employed. The results also confirm the effectiveness of the proposed error-recovery scheme in recovering from CDC failures.**

***Index Terms*—Clock domain crossing, error recovery, fault detection.**

I. Introduction

SYSTEM-ON-CHIP integrated circuits today offer diverse functionality and contain billions of transistors. However, high-speed communication between cores remains a major challenge. This problem is exacerbated when cores operate in separate clock domains and at different clock frequencies.

In multiclock designs, a clock-domain crossing (CDC) occurs whenever data is transferred between clock domains. Depending on the relationship between the sender and receiver clocks, various types of problems may arise during data transfer. Propagation of metastability, data loss, and data incoherency are three fundamental problems of multiclock design, all of which are caused by CDC faults [2].

To reduce the probability of propagating metastability through the design, designers employ synchronizers at clock boundaries. Moreover, to avoid data loss and to ensure proper transmission and reception of data in multiclock designs, designers also rely on appropriate CDC protocols. Data incoherency, which mainly occurs where CDC signals reconverge, is avoided by making designs tolerant of the variable delays that occur on reconvergent paths [3]. Verification techniques and commercial verification tools enable designers to check designs for CDC-associated problems and verify the correctness of functional behavior [4]–[6]. If CDC errors are not addressed early in the design cycle, many chips are likely to exhibit functional errors during post-silicon validation. To address the metastability that occurs in multiclock circuits, and consequently to increase the mean time between failures (MTBF), designers typically employ different types of synchronizers, among which the most commonly used is a pair of flip-flops residing on the clock boundaries.

As we move toward higher integration levels and even smaller technology nodes, errors that occur due to process variations, design marginalities, and corner operating conditions are starting to play a more important role in multiclock circuits. Consequently, circuits that were deemed to be fault free through CDC analysis during presilicon validation may exhibit CDC errors after fabrication.

Therefore, the effect of process variations on correct operation of multiclock circuits must be investigated, and there is a need for testing techniques for CDC faults. A test-pattern selection method, for detecting CDC faults, was recently proposed in [7]. A commercial ATPG tool and a commercial logic simulator were used to extract, from a pattern repository, a set of test patterns that detect CDC faults. However, repeated invocation of the simulator leads to long runtimes. Moreover, the tests derived in [7] do not target at-speed transfer of transition of data required between the clock domains; hence, their effectiveness for high-speed circuits is questionable.

In this paper, we focus on testing of CDC faults and, in particular, we integrate solutions for detecting and locating CDC faults, and ensuring post-silicon recovery from CDC failures. The contributions of this paper, which include a complete framework to detect, diagnose, and recover from CDC failures, are:

1) an automatic test-pattern generation (ATPG) method based on bounded time-frame expansion and logic constraints;
2) a fault diagnosis method to locate CDC faults considering the relative clock frequencies of different clock domains;

Manuscript received October 13, 2012; revised December 27, 2012 and March 5, 2013; accepted March 11, 2013. Date of current version August 16, 2013. This work was supported in part by the National Science Foundation under Grant CCF-0903392 and SRC under Contract 1992. A preliminary version of this paper was presented at the 2012 IEEE/ACM Design, Automation, and Test Conference in Europe [1]. This paper was recommended by Associate Editor X. Wen.

N. Karimi was with Duke University, Durham, NC 27705 USA. She is now with the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY 11201 USA (e-mail: nkarimi@poly.edu).

K. Chakrabarty is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: krish@ee.duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2013.2255127

3) an error recovery scheme to handle CDC failures;
4) detailed HSpice simulations to validate the proposed error-recovery scheme;
5) a comprehensive set of results to demonstrate the effectiveness of the proposed CDC fault detection, diagnosis, and error-recovery schemes.

The remainder of this paper is organized as follows. Section II discusses methods used to resolve metastability. Section III discusses the need for post-silicon CDC testing. In Section IV, we introduce the CDC fault models to represent the faulty behavior of CDC designs in the presence of physical defects. Section V presents the ATPG algorithm targeting CDC faults. In Section VI, first we describe our fault diagnosis method, and based on this technique, we present an error-recovery scheme to tolerate CDC failures. Results demonstrating the effectiveness of the proposed frameworks are presented and discussed in Section VII, and conclusions are drawn in Section VIII.

II. Resolving Metastability

Synchronizers are used to mask the effect of metastability in multiclock circuits [3]. It is expected that in a design, including synchronizers, the output of a flip-flop rarely becomes metastable, e.g., only once in every MTBFs years, typically, 20 years for clock frequencies of 400 MHz [8]. However, for faster clocks, the probability of observing metastability at the outputs of flip-flops increases rapidly, e.g., the MTBF drops to 1 min for a clock frequency of 1 GHz [8].

To prevent incorrect operation due to metastability, both asynchronous and synchronous handshaking mechanisms between different clock domains have been proposed in the literature. In the asynchronous handshaking mechanism, a request is first sent from the sender to receiver domain. After sending the request, the sender sends the data to the receiver. The receiver sends out acknowledgement to the sender to indicate completion of data transfer. Upon receiving the acknowledgement, the sender can send another request to the receiver. To immunize the handshaking mechanism against the metastability of the request and acknowledge signals, synchronizer flip-flops are inserted in the circuit [9].

Although an asynchronous handshaking method is immune to CDC faults, it suffers from uncertainly and indeterministic delay of data transfer between different domains. To achieve higher performance, FIFOs (and particularly two-clock FIFO synchronizers) are used in multiclock circuits. However, the size of the FIFO buffers is a concern and what size FIFO to use can be a difficult design decision. The larger a FIFO is, the higher is the cost [8].

Synchronizers without handshaking allow us to overcome the drawbacks of asynchronous handshaking in the transfer of data between different domains. The use of two flip-flop synchronizers is common in multiclock circuits [8]. However, fast clocks, low supply voltages, and extremely low or high temperatures decrease MTBF and necessitate the use of additional synchronizer flip-flops. To decrease MTBF in such cases, four flip-flop synchronizers may be used in clock boundaries [8].

The flip-flops used as synchronizers must be more robust to variations in process, temperature, and voltage. Ideally, the setup and hold time of synchronizer flip-flops should be zero. However, it is costly to use synchronizer flip-flops with negligible setup and hold time. For example, a nearly-zero setup time flip-flop presented in [10] requires 66% area overhead compared to a typical flip-flop.

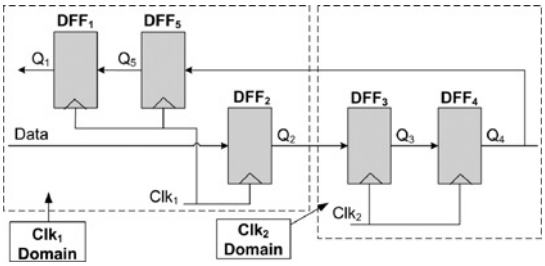

Fig. 1. Generic CDC circuit [13].

In state-of-the-art SoCs, thousands of bits of data are transferred between different clock domains [11]. Due to the timing uncertainty of asynchronous handshaking as well as the high cost associated with the use of special synchronizer flip-flops with zero setup times, it is more practical for multiclock SoCs to use typical synchronizer flip-flops to transfer data between clock domains. To control the clocking of different domains in the multiclock circuits, equipped with synchronizers in clock boundaries, and to avoid setup and hold-time violations, a number of dummy cycles are added to each clock domain. These dummy cycles control the skew required between different clock signals. The number of inserted dummy cycles depends on the relative phase and frequency of clock signals in different domains.

III. Impact of Process Variation on CDC Faults

The motivation for our work lies in our observation that multiclock circuits, even when equipped with synchronizers at clock boundaries, may exhibit incorrect behavior due to process variation-induced violation of setup and hold time at the boundary flip-flops.

In reality, the parameters of fabricated transistors do not always match design specifications due to process variations. These variations directly result in deviations in transistor parameters, such as threshold voltage, oxide thickness, and W/L ratios, and significantly impact the functionality of circuits [12].

To evaluate the impact of random process variations on the transfer of data between different clock domains, even when synchronizer flip-flops are employed at clock boundaries, we conducted a series of HSpice simulations under process variations for a generic CDC circuit, shown in Fig. 1. In this circuit, flip-flops DFF_2 and DFF_4 reside in different clock domains and act as sender and receiver flip-flops, respectively. Flip-flop DFF_3 is employed as a synchronizer.

To determine the effect of random process variation in the transfer of data between clock domains, we ran several HSpice Monte Carlo (MC) simulations on the circuit, shown in Fig. 1, using the 45-nm predictive technology model [14]. Simulations were carried out using the following process-variation parameters for a Gaussian distribution: transistor gate length L: $3\sigma = 10\%$; threshold voltage V_{TH}: $3\sigma = 30\%$, and gate-oxide thickness t_{OX}: $3\sigma = 3\%$. The process variation data reflects a 45-nm process in commercial use today. First, to isolate the effect of process variation on data transfer between different clock domains of the circuit, shown in Fig. 1, and to

TABLE I
NUMBER OF SETUP TIME VIOLATIONS FOR DIFFERENT NUMBERS OF MONTE CARLO (MC) SIMULATIONS

Total no of MC runs	No. of runs with setup time violation (%) assuming Gaussian parameters for DFF_3	No. of runs with setup time violation (%) assuming Gaussian parameters for all flip-flops
2000	1009 (50.4%)	1016 (50.8%)
4000	2061 (51.5%)	1970 (49.2%)
6000	3061 (51%)	2902 (48.4%)
8000	4050 (50.6%)	3893 (48.7%)
10000	5033 (50.3%)	4899 (49%)

show incorrect behavior of the circuit due to process variation-induced violation of setup time at the boundary flip-flops, only the parameters for flip-flop DFF_3 are assumed to have a Gaussian distribution, and the parameters for the other four flip-flops are assigned deterministic values.

We recorded the number of experiments in which the setup time of the flip-flop DFF_3 were violated under process variations. The results are shown in the second column of Table I. We found that in more than 50% of the experiments, variations of the parameters of DFF_3 result in a setup time violation at the receiver flip-flop, and consequently in incorrect circuit operation even when synchronizer flip-flops are employed. Similar results (third column of Table I) were obtained when we considered the same process variation model for all the flip-flops in Fig. 1. These results highlight the fact that due to the effect of process variations, design verification does not accurately predict silicon behavior for clock domain crossings and synchronizers do not prevent errors; therefore, manufacturing testing for CDC faults is necessary.

Transition delay fault (TDF) testing is widely used in industry to target timing-related defects. Despite their benefits, current transition ATPG tools are not adequate for detecting CDC faults because these tools do not model and target the interaction between logic residing at clock boundaries when test patterns are generated for TDFs. Path-delay test methods [15] suffer from the scalability problem for large designs, and the timing-critical paths that they target do not necessarily include clock-domain crossings. We show in this paper that TDF test patterns are not adequate for CDC faults, and they lead to a coverage gap. Therefore, fault models, ATPG methodologies, and diagnosis and recovery schemes need to be developed to specifically target CDC faults.

IV. CDC FAULT MODEL

To be able to screen CDC defects, the faulty behavior of these defects must be logically represented using a fault model. In this section, we review a fault model that was presented in [7].

In a synchronous circuit, the proper operation of a flip-flop depends on the stability of its input signal for a certain period of time before (setup time) and after (hold time) its clock edge. If setup and hold times are violated, the flip-flop output may oscillate for an indefinite amount of time, and may or may not settle to a stable value before the next active clock edge. This unstable behavior is known as metastability. Fig. 2(a) shows an example of a multiclock circuit in which signal S is launched by Clk_1, and needs to be captured properly by

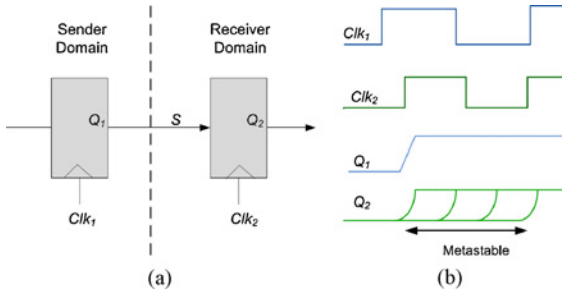

Fig. 2. Example of a CDC circuit and metastability. (a) CDC circuit. (b) Metastability on Q_2.

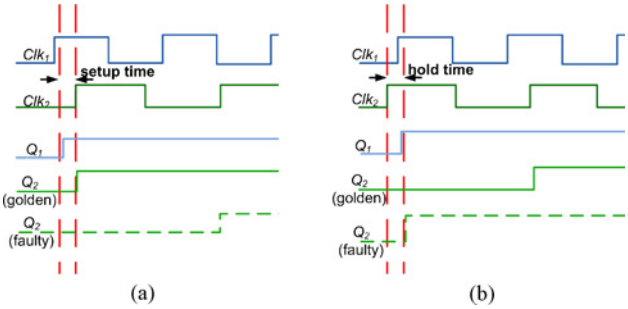

Fig. 3. Timing waveforms showing setup and hold-time violations for the circuit in Fig. 2(a). (a) Setup-time violation. (b) Hold-time violation.

Clk_2. As shown in Fig. 2(b), if a transition on S happens very close to the active edge of Clk_2, a setup-time violation occurs, which may lead to metastability on Q_2.

CDC faults mainly occur due to setup and hold-time violations on flip-flops residing at clock boundaries. If a flip-flop experiences a setup-time violation, it does not sample a change in value at its data input. In a hold-time violation, however, it may incorrectly capture a data change at its input. We next describe the fault model for each case.

A. Setup-Time Violation

Fig. 3 illustrates sample waveforms for the CDC circuit of Fig. 2(a). As shown in Fig. 3(a), if signal S experiences an unexpected delay and its value changes during the setup-time window of the receiver flip-flop, the receiver flip-flop may capture the value 0 even though the expected value is 1. Since the output of the sender flip-flop does not change in the subsequent clock cycle, Q_2 gets its expected value of 1 in the next clock cycle. In this case, the setup-time violation of the receiver flip-flop can be modeled as a slow-to-rise fault with a delay of one clock cycle. However, if the width of the transition on the output of the sender flip-flop is not long enough, the receiver flip-flop will not capture that transition, and remains unchanged. In this case, the setup-time violation of the receiver flip-flop can be modeled by a slow-to-rise fault with infinite delay. In practice, safe passage of one CDC signal between two clock domains through a two-flip-flop synchronizer requires that the CDC signal be 1–1.5 times wider than the receiver clock period [16].

In general, if a value change of a CDC signal S violates the setup time of the receiver flip-flop, then the faulty behavior can be modeled as a transition (slow-to-rise or slow-to-fall) fault with a delay of k clock cycles, where $k = 1$ if the pulse observed in signal S is at least 1.5 times wider than the receiver

clock period. Otherwise, $k = \infty$. In the rest of this paper, a CDC fault arising due to setup-time violations will be referred to as a S-CDC fault.

B. Hold-Time Violation

If a flip-flop experiences a hold-time violation, data changes on its input may be incorrectly sampled. Fig. 3(b) shows another sample waveform for the CDC circuit of Fig. 2(a). If signal S changes during the hold-time interval of the receiver flip-flop, an incorrect change on the output may be observed. The receiver flip-flop gets an output value of 1 one clock cycle earlier than expected. In this case, the hold-time violation at the receiver flip-flop can be modeled as a transient fault with a duration of one clock cycle. Similarly, if the output of the sender flip-flop changes before the next active edge of the receiver flip-flop, the receiver flip-flop captures the transition of signal S, and the hold-time violation of the receiver flip-flop can be modeled as a transient fault with a duration of one clock cycle. H-CDC faults used to refer to the CDC fault arising due to hold-time violations. In this paper, we focus on S-CDC faults and leave the treatment of hold-time violations for future work.

V. Fault Detection Method

A TDF ATPG tool cannot be used to detect all S-CDC faults. It typically launches a transition at the fault site and propagates it to an observable output, i.e., either a scan flip-flop or a primary output. While these steps are also necessary to detect S-CDC faults, they are not sufficient. The detection of S-CDC faults requires fault excitation and propagation through paths from the sender domain. However, this requirement is not always met when TDF ATPG tools are used for test generation.

Launch-on-shift (LoS) and launch-on-capture (LoC) are two widely used TDF testing methods. In LoS, the second pattern of a two-pattern test is obtained by a one-bit shift of the first pattern. However, in the LoC scheme, the second pattern is obtained from the circuit response to the first pattern. Although LoS usually provides higher delay-fault coverage and offers ease of test-generation compared to LoC, it requires significant design effort to achieve at-speed switching of the scan-enable signal. Therefore, due to the area overhead and design-time overhead of the LoS method, LoC is preferred to LoS [17]. In this paper, we only consider LoC for detecting S-CDC faults.

A. Test Generation Process

In this section, we discuss our test-pattern generation method, which is referred to as CDC-oriented triple-capture (CoTC). To describe the testing method to detect S-CDC faults, we use the simple multiclock domain circuit, shown in Fig. 4. In this circuit, for the sake of clarity, only the flip-flops at clock boundaries are shown. Note that throughout this paper, we consider a single-fault model.

In this paper, no assumptions are made or restrictions are placed on the clocking scheme. The clock signals are fed either by different PLL sources, or by a common PLL source but with different phases and frequencies. We assume that the frequency of the clock signal of the sender (receiver) domain is an integral multiple of the clock frequency of the receiver

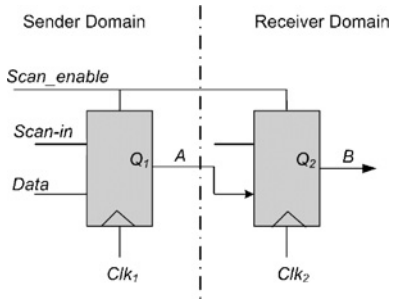

Fig. 4. CDC example for illustrating the proposed ATPG method.

(sender) domain. Accordingly, the phase difference between sender and receiver clocks may not lead to any setup and hold-time violation problem if there is no such violation in the first few clock cycles. To resolve the violation that may occur in the first few clock cycles due to the small related phase of sender and receiver clocks, the use of conflict detectors have been proposed in literature [8]. A conflict detector identifies when the sender and receiver clocks are dangerously close to each other. In the case of imminent problem, the clock signal of the receiver domain is delayed.

Assume that we want to target the S-CDC fault modeled by a slow-to-rise fault at the output of the receiver flip-flop (signal B) in the circuit, shown in Fig. 4. To detect this fault, first a rising transition must be generated on A, and then this transition must be propagated to B in the next active edge of Clk_2. Note that the transitions on A and B must be at-speed with respect to Clk_1 and Clk_2, respectively.

The clock frequencies of the sender and receiver domains, F_S and F_R, respectively, must be considered in CoTC to generate test-patterns targeting S-CDC faults. We assume that these frequencies are specified by the designer, and therefore are known during test-pattern generation. We next describe the steps for each case. In each step, A and B keep their values, unless otherwise mentioned. Note that, in this paper, we consider separate scan-chain for each clock-domain. To apply detection, diagnosis, and recovery procedures for CDC faults, we merge all the scan-chains by connecting the scan-out of each chain to the scan-in of another chain. A small amount of multiplexing is assumed so that the scan-in and scan-out signals can be kept separate if the clock domains are to be tested separately for intra-domain faults. The hardware overhead is negligible because the multiplexing is done only for the scan signals and not for the functional I/Os. In addition, test-mode and test-clock input pins of each scan-chain are fed by the common test-mode and test-clock signals, respectively.

1) *Case 1: $F_S = F_R$:* The first case deals with test-pattern generation for multiclock circuits in which the flip-flops residing in sender and receiver boundaries operate at same clock frequency, i.e., $F_S = F_R$. In this case, to ensure an at-speed transition on A with respect to Clk_1, and an at-speed transition on B with respect to Clk_2, we need to apply four test vectors instead of the two that are applied by the traditional LoC method. Steps 2 and 3 ensure that the transitions on A and B are at-speed with respect to Clk_1 and Clk_2, respectively. Fig. 5(a)–(d) shows the active paths highlighted in bold for the four steps needed to detect the CDC fault.

The four steps in CoTC to target the S-CDC fault modeled by a slow-to-rise fault on B are as follows.

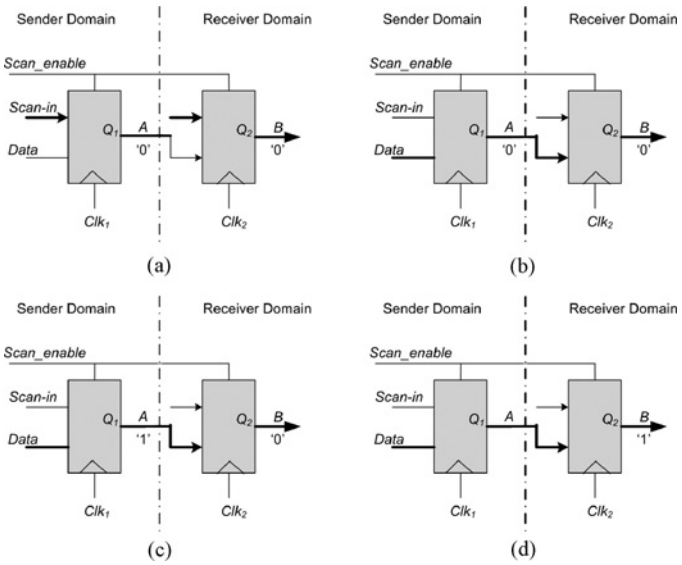

Fig. 5. Illustration of all steps to target slow-to-rise S-CDC fault on signal B (active path highlighted in bold). (a) Step 1. (b) Step 2. (c) Step 3. (d) Step 4.

1) *Step 1)*: Shift vector V_1 to the circuit in scan mode such that A and B both get the value 0 in this step.
2) *Step 2)*: Switch to functional mode and generate vector V_2 such that A and B are both 0.
3) *Step 3)*: Operate in functional mode and generate vector V_3 such that in this step, the values on A and B are 1 and 0, respectively. This step ensures that a transition is launched at-speed across the CDC.
4) *Step 4)*: Operate in functional mode and generate vector V_4 such that B gets the value 1.

If the flip-flops residing in sender and receiver boundaries operate at the same clock frequency, the S-CDC fault modeled by a slow-to-rise fault on signal B can be detected by applying vectors V_1 to V_4 (as discussed above) in four consecutive clock cycles. During scan mode (Step 1), a common shift clock signal is applied to both sender and receiver domains but in Steps 2–4; the circuit operates in functional mode and we apply Clk_1 and Clk_2 to the first and second clock domains, respectively. Note that each of vectors V_1 to V_4 includes two parts; the first part includes the values of the flip-flops and the second part includes the values of the primary inputs of the circuit in each step.

2) *Case 2: $F_R = M \cdot F_S$:* In this case, the frequency of functional clock Clk_2 is an integer multiple of the frequency of functional clock Clk_1.

To target the S-CDC fault modeled by a slow-to-rise fault on B of Fig. 4, first a rising transition must be generated on A, and then this transition must be propagated to B in the next active edge of Clk_2. The transitions on A and B must be at-speed with respect to Clk_1 and Clk_2, respectively. Therefore, to generate test-patterns to detect such faults, the following steps are necessary.

1) *Step 1*: Shift a vector to the circuit in scan mode such that A and B both get the value 0 in this step.
2) *Step 2*: Switch to functional mode and apply one functional clock cycle using Clk_1 and M functional clock cycles using Clk_2. A and B should get the value 0 in these clock cycles. This constraint is ensured using a justification procedure. Note that in this case $F_R = M \cdot F_S$, and therefore while an at-speed transition is generated on A with respect to Clk_1, M clock cycles using Clk_2 are applied to the circuit as well.
3) *Step 3*: Operate in functional mode and apply one functional clock cycle using Clk_1 and one functional clock cycle using Clk_2. In this step, the values on A and B should be 1 and 0, respectively (ensured via justification).
4) *Step 4*: Operate in functional mode and apply one functional clock cycle using Clk_2. B should get the value 1 in this step.

3) *Case 3: $F_S = N \cdot F_R$:* The third case occurs when the sender domain operates N times faster than the receiver domain, where N is an integer. Similar to the previous cases, to detect the slow-to-rise S-CDC fault on B of Fig. 4, first a rising transition must be generated on A, and then this transition must be propagated to B in the next active edge of Clk_2. As noted above, the transitions on A and B must be at-speed with respect to Clk_1 and Clk_2, respectively. The steps taken in this case are as follows.

1) *Step 1*: Shift a vector to the circuit in scan mode such that A and B both get the value 0 in this step.
2) *Step 2*: Switch to functional mode and apply $N - 1$ functional clock cycles using Clk_1 and one functional clock cycle using Clk_2. A and B should get the value 0 in these clock cycles.
3) *Step 3*: Operate in functional mode and apply one functional clock cycle using Clk_1. In this step, A should get the value 1.
4) *Step 4*: Operate in functional mode and apply one functional clock cycle using Clk_2. B should get the value 1 in this step.

Note that in all the cases discussed above, Step 2 ensures an at-speed transition on signal A. In practice, if A does not drive any logic in the sender domain, any delay-fault that leads to a delayed transition on A will not be detected if Step 2 is not taken.

In all the cases discussed above, if there is combinational logic between sender and receiver flip-flops, the test generation condition is changed based on the type of the gate at the clock boundary. However, in practice, synchronizer flip-flops are always embedded at the clock boundaries. Accordingly, in this paper, we considered a direct connection between sender and receiver flip-flops. On the other hand, since the clock frequency of sender or receiver domains are not always equal (case 2 and case 3), path-delay test methods are not adequate for detecting CDC faults.

B. Test Application Procedure

To test a multiclock circuit using the test patterns generated by CoTC, the relative frequencies of sender and receiver domains should be considered. Similar to the test generation process that was discussed in Section V-A, based on the values of F_S and F_R, different cases may arise for applying the CoTC patterns. In this section, we discuss the case where the sender and receiver domains operate at the same clock frequencies. Other cases can be treated using a similar procedure.

1) *Case 1: $F_S = F_R$:* To test such circuits using the CoTC test patterns, the following steps should be taken.

a) *Step 1*: Set the circuit to scan mode. Scan in the initialization vector (V_1), and set the values on primary inputs.

b) *Step 2*: Switch to functional mode. Insert dummy cycles if needed to give scan-enable (SE) time to flip. Operate in functional mode and apply three functional clock cycles using Clk_1 and three functional clock cycles using Clk_2. Recall that we applied a total of three functional clock cycles using Clk_1 and three functional clock cycles using Clk_2 during test-pattern generation for this case (Steps 2-4 of Case 1 in Section V-A).

c) *Step 3*: Switch to scan mode and shift out the results. This step can be overlapped with Step 1 to apply another test-pattern to the circuit.

C. CoTC Implementation Details

To implement CoTC, we leveraged a commercial ATPG tool. First, full-scan insertion was performed. Next, pairs of flip-flops residing in clock boundaries (in different clock domains) were extracted. Finally, test generation was performed under the constraints discussed in Section V-A.

In this section, we deal with the implementation details of CoTC when the sender and receiver domains operate at the same clock frequencies (Case 1 in Section V-A). For the other two cases (Case 2 and Case 3), CoTC can be implemented in a similar manner.

Consider the case where the sender and receiver flip-flops operate at the same clock frequency (with same or different phases). In this case, CoTC requires that the CDC flip-flops get specific values in four consequent clock cycles. However, commercial ATPG tools cannot be directly used to generate test patterns such that all of these requirements are met simultaneously. Therefore, to generate test patterns that satisfy the CoTC requirements for a S-CDC fault, we first expand the circuit in time, and then use a commercial ATPG tool to generate test patterns targeting that fault in the time-expanded model of the circuit.

To implement CoTC with one launch and three capture cycles, we triplicate the combinational logic of the circuit under test, and then use the triplicated version of the circuit for test generation. The values that should be considered for each pair of boundary flip-flops in four consecutive clock cycles in CoTC, provided as constraints for each time frame.

Fig. 6(a) shows an example CDC circuit under test. In this figure, C_1 and C_2 are combinational blocks. A CDC path exists between flip-flop DFF_1 and flip-flop DFF_2. Therefore, to apply CoTC to detect the S-CDC fault modeled by a slow-to-rise fault on the output of DFF_2, we must ensure that the outputs of DFF_1 and DFF_2 get the values 00, 00, 10, and X1 in four consecutive clock cycles, respectively. Note that X refers to a don't care.

Fig. 6(b) illustrates the triplicated time-expanded model of the circuit shown in Fig. 6(a). In this figure Q_1, $Q_{1,1}$, $Q_{1,2}$, and $Q_{1,3}$ represent the output of flip-flop DFF_1 in consecutive clock cycles. Similarly, the output of flip-flop DFF_2 in consecutive clock cycles is denoted by Q_2, $Q_{2,1}$, $Q_{2,2}$, and $Q_{2,3}$. Table II shows the values required at Q_1 and

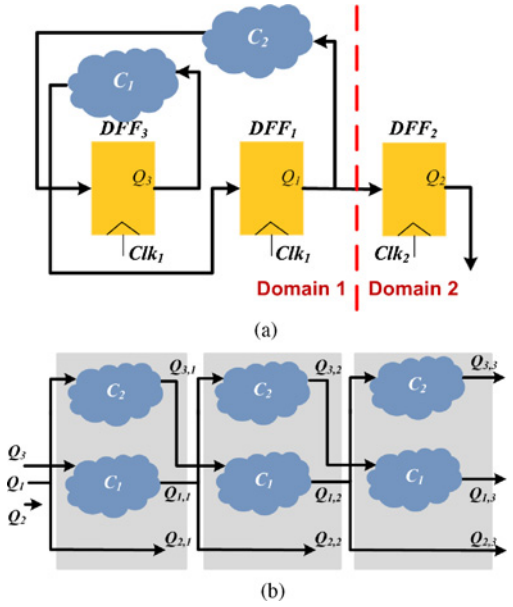

Fig. 6. Example of a CDC circuit and its expanded model. (a) Example of a CDC circuit under test. (b) Expanded model (three time frames) of the CDC circuit.

TABLE II
VALUES AT FLIP-FLOPS REQUIRED FOR DETECTING A SLOW-TO-RISE S-CDC FAULT ON Q_2 USING COTC

Q_1	Q_2	$Q_{1,1}$	$Q_{2,1}$	$Q_{1,2}$	$Q_{2,2}$	$Q_{1,3}$
0	0	0	0	1	0	X (don't-care)

Q_2 in the three time frames for CoTC to detect the slow-to-rise S-CDC fault on Q_2. The values shown in this table must be observed in the same clock cycle in the pseudo-combinational (expanded) model of the circuit under test. Therefore, we can use a commercial ATPG tool to generate test patterns to detect the stuck-at 0 fault on $Q_{2,3}$ in Fig. 6(b), while considering the values, shown in Table II, as ATPG constraints. We maintain a scoreboard for each S-CDC fault and the test vectors that detected that fault. Finally, a minimum set covering algorithm is used to select a minimal set of vectors that detect all slow-to-rise S-CDC faults.

VI. FAULT DIAGNOSIS AND RECOVERY

If a CDC fault is detected, post-silicon fault diagnosis and error recovery must be initiated to ensure correct operation. Fault diagnosis is necessary for the identification of manufacturing defects, and accordingly speeding-up yield ramp-up. Information provided by the diagnosis process is used in the physical inspections of the circuit. During the failure analysis process, it is important to locate the cause of failures quickly and accurately. Fault location may be required to analyze the defect causing the faulty behavior, reconfigure the circuit to mask the faulty behavior of the circuit, or replace the faulty subcircuit [18], [19].

A. Proposed Fault Diagnosis Method

Fault diagnosis methods can be categorized into two groups: cause–effect and effect–cause approaches [20]. In cause–effect

methods, a fault dictionary is used for fault location. Effect–cause methods do not need a fault dictionary. These methods start from faulty outputs of the circuit under test and reason back through the logic to identify possible fault candidates. In this paper, we propose a cause–effect approach for the diagnosis of S-CDC faults, since it is potentially faster if a compact dictionary can be generated.

Locating a fault using a fault dictionary requires applying the vectors included in the fault dictionary to the circuit-under-test (CUT) and comparing the responses of the observable outputs with the values stored in the fault dictionary. Full-dictionaries include the response of CUT to a given test set in the presence of each fault. Although fault diagnosis methods that use full-dictionaries provide high resolution, these methods suffer from the large size and high generation time of fault dictionaries [21].

To overcome the above problem, pass–fail dictionaries have been proposed in the literature [22]. A pass–fail fault dictionary contains a single bit for each fault F and test vector TV pair. This bit shows whether fault F is detectable by applying test vector TV to the CUT. For large circuits, pass–fail dictionaries are preferred to full-dictionaries, even at the expense of some degradation in fault resolution.

1) *Fault Dictionary Design:* The proposed fault dictionary includes a set of test patterns, a signature of the expected response of the CUT to each test pattern, and the CDC faults that can be detected by each pattern. Obviously, this dictionary is smaller than a full-dictionary that includes the response of the CUT to each test pattern in the presence of each fault.

To generate the CDC-fault dictionary, the following steps should be taken.

a) *Step 1*: First, CoTC is applied to the CUT and up to 255 test patterns are generated for each detectable S-CDC fault. Although this method is general for any number of test patterns, 255 was deemed to be sufficient in our work. Set P_i ($1 \leq i \leq N$, N: number of S-CDC faults) includes all patterns generated by CoTC to detect S-CDC fault f_i.
b) *Step 2*: A subset of the patterns generated in Step 1 are selected such that by using the selected patterns, any two S-CDC faults f_i and f_j are distinguishable from each other. In this step, $P_{i,j}$ is generated for each pair of faults f_i and f_j and includes all the patterns generated by CoTC detecting exactly one among f_i and f_j.
c) *Step 3*: In this step, a minimum set covering algorithm is applied to the set of test vectors generated in Step 2 for each pair of S-CDC faults to select a minimal set that distinguishes all S-CDC faults from each other. These patterns are stored in the CDC-fault dictionary.
d) *Step 4*: For each test pattern selected in Step 3, the expected response of the CUT is determined by logic simulation. The response includes the values of all observable points, including primary outputs and scan flip-flops.
e) *Step 5*: To reduce the storage required for the expected response of the CUT for each test pattern (evaluated in Step 4), a signature of the expected values of primary outputs and flip-flops is extracted and stored in the distinguishable dictionary along with each test pattern.
f) *Step 6*: Along with each test pattern and the expected response of the CUT to that pattern, a list of S-CDC faults that can be detected by that pattern is stored in the CDC-fault dictionary.

As discussed in Step 5, to reduce the size of the CDC-fault dictionary, instead of expected outputs of the CUT to each test pattern, a signature of those values are stored. We use a 64-bit cyclic redundancy check (CRC) code for response compaction and encode the sequence of primary outputs and the sequence of flip-flip outputs related to each test pattern, separately. The signatures and their related test patterns are stored in the CDC-fault dictionary.

2) *CDC Fault Diagnosis:* Using the fault dictionary generated by the method discussed in the previous section, all detectable CDC faults can be located. The CDC-fault dictionary generated for each circuit includes a number of test patterns (values that should be applied to the primary inputs, and initial state of flip-flops) along with a signature of expected values of the observable points of the circuit (primary outputs and scan flip-flops), while applying each test pattern to the circuit and the list of CDC faults that can be detected by applying each test pattern.

To locate a CDC fault, the clock frequencies of sender and receiver domains should be considered. Based on the values of F_S and F_R, different cases may occur. We discuss below the case where both sender and receiver domains operate at the same clock frequency. Other cases can be treated similarly (as in Section V).

To locate a CDC fault, the test patterns included in the generated fault dictionary should be applied to the CUT, one after another, until the exact location of that CDC fault is diagnosed or no other test pattern is left in the fault dictionary. Before applying the fault diagnosis algorithm to the CUT to locate a CDC fault, all of the detectable CDC faults are included in the suspect list and are considered as the candidate locations. Then the following steps are taken while applying each test pattern included in the fault dictionary to the CUT.

a) *Step 1*: Set the circuit to scan mode. Scan in the initialization vector (V_1), and set the values on primary inputs.
b) *Step 2*: Switch to functional mode. Insert dummy cycles if needed to give Scan enable (SE) time to flip. Operate in functional mode and apply three functional clock cycles using Clk_1, and three functional clock cycles using Clk_2.
c) *Step 3*: Switch to scan mode and shift out the results. If the signature of the results matches the expected signature included in the fault dictionary for this test pattern, delete all the faults that are diagnosable by this test pattern from the list of suspect locations. This step can be overlapped with Step 1 to apply another test-pattern to the circuit.

As discussed above, all the test patterns included in the CDC fault dictionary are applied to the CUT, one after another. While applying each test pattern, if the results matches the expected results, the faults listed as diagnosable by that test pattern are excluded from the list of suspect faults. Note that in this section, we discussed the case where the sender and receiver domains operate at the same clock frequencies. Other cases can be treated using a similar procedure.

In principle, using the proposed fault diagnosis method, all S-CDC faults are distinguishable from each other and the exact location of each S-CDC fault can be determined. However, due to the limitation of commercial ATPG tool that we employ in this paper, only a subset of the test patterns detecting each CDC fault (up to 255 patterns) is generated for that fault (Section VI-A). Due to this limitation, for a number of CDC faults, their exact location cannot be determined and instead, a list of suspect locations is reported. As an example, assume that set TV_i and set TV_j includes the set of test patterns generated by a commercial ATPG tool to detect CDC faults f_i and f_j, respectively. The sets TV_i and TV_j do not include all the patterns detecting faults f_i and f_j, i.e., each of TV_i and TV_j sets includes up to 255 test patterns. Although both these faults can be detected by test pattern t_k, due to the limitation of the commercial ATPG tool, t_k may only be included in TV_i (not in TV_j). Hence, even though CDC faults f_i and f_j are considered as distinguishable by applying vector t_k, they cannot be distinguished from each other on the basis of t_k.

B. Error Recovery

To recover from errors that result from process variations, the use of post-silicon tunable-buffers has been proposed in the literature [23], [24]. These buffers can compensate for the effect of process variations. We consider such an approach to recover from CDC errors.

1) *Proposed CDC Error Recovery Method:* As discussed in Section III, process variations may result in an incorrect transfer of data between different clock domains of a multiclock circuit. Equipping multiclock chips with clock-tuning circuits can enhance the reliability of these circuits and compensate the effect of process variations [25], [26].

As discussed in Section IV, if the setup time of a flip-flop is violated, its faulty behavior can be modeled as a transition fault. Accordingly, to recover from the erroneous behavior of a flip-flop when its setup time is violated, its clock signal can be delayed.

To recover a multiclock circuit from a S-CDC error, the receiver flip-flop of the faulty CDC pair should operate under a delayed clock signal. Therefore, external delay blocks can be inserted in the clock path of such a flip-flop depending on the slack-time between it and the flip-flops fed by it. Fig. 7(a) shows an example multiclock circuit in which the flip-flop residing in the receiver clock boundary operates under a delayed-clock signal. In this circuit, by inserting a buffer in the clock path of the receiver flip-flop (depending on the propagation delay of BUF_1 and the amount of setup-time violation of that flip-flop), S-CDC errors in the clock boundary can be avoided.

In general, to equip a multiclock circuit with a CDC error recovery mechanism, the circuit shown in Fig. 7(b) can be employed. If by applying the fault diagnosis scheme proposed in Section VI-A, the pair of flip-flops shown in Fig. 7(b) is reported as being faulty, A is set to value 1, and accordingly, Clk_2 signal propagates through gate BUF_1. Otherwise, A gets the value 0. As shown in this figure, to retain the timing relationship between Clk_1 and Clk_2, another tri-state buffer is inserted in the Clk_1 path.

The circuit shown in Fig. 7(b) includes one flip-flip in the receiver side of the clock boundary. To equip this circuit

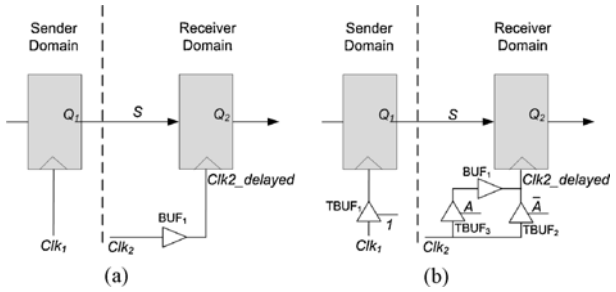

Fig. 7. Example of a CDC circuit (a) with delayed receiver-clock signal and (b) equipped with an error recovery mechanism.

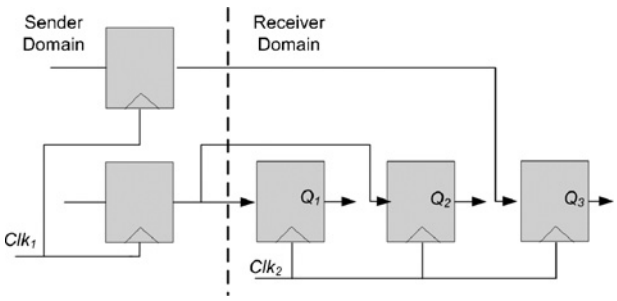

Fig. 8. Example of a CDC circuit with three flip-flops in the receiver clock-boundary.

with an error-recovery mechanism, one buffer, one inverter (to generate $\overline{A}$) and two tri-state buffers are inserted in the receiver domain. In addition, one tri-state is inserted in the clock path of the sender flip-flop. If the receiver domain includes m flip-flops out of which n flip-flops reside in the clock boundary, the error-recovery circuitry includes n buffers, $2n$ tri-state buffers, and n inverters. In addition, to reduce the number of input pins added to the original multiclock circuit, one shift register (including $\lceil \log_2(n+1) \rceil$ registers) and one decoder (with $\lceil \log_2(n+1) \rceil$ inputs) are also employed. One tri-state buffer is inserted in the sender domain and it feeds the clock input of all the flip-flops residing in this domain. Another tri-state buffer is located in the receiver domain feeding the clock input of all flip-flops other than those reside in the clock boundary.

Fig. 8 shows another example of a two-clock domain circuit that includes three flip-flops in the receiver side of the clock boundary and two flip-flops in the sender side of the clock boundary. In this figure, for the sake of clarity, only the flip-flops in the clock boundaries are illustrated and the other flip-flops have not been shown. Fig. 9 shows this circuit after insertion of the proposed error-recovery hardware. As shown in Fig. 9, three out of four outputs of the inserted decoder are connected to the clock circuitry of the receiver domain. The other output of decoder (A_0) is enabled, while the circuit is fault free and no CDC fault is diagnosed. Exactly, one of the three signals A_1-A_3 get the value of 1 if one of the three flip-flops in the receiver clock-boundary experiences setup-time violation and needs to be operated under a delayed clock signal. The faulty flip-flop residing in the receiver clock-boundary is identified by the fault dictionary generated using the method discussed in Section VI-A. This encoded value is fed to the error recovery circuitry through the *Data-in* signal.

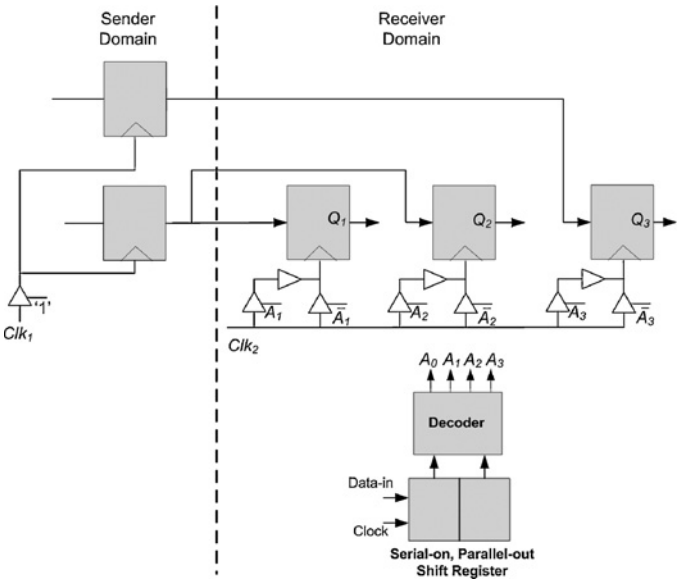

Fig. 9. Error-recoverable model of the circuit shown in Fig. 8 with one delay buffer in the clock path of the faulty flip-flop.

When error recovery is employed, the clock signal of the faulty flip-flop is delayed for d ns, where d is the propagation delay of one buffer gate. We can easily extend the proposed scheme and add more buffers in the clock path of a faulty flip-flop when insertion of only one buffer is not sufficient to recover from the CDC error occurred due to the setup time violation of that flip-flop.

Fig. 10 shows the CDC error-recoverable implementation of the circuit shown in Fig. 8 with the capability of adding up to four buffers in the clock path of the faulty flip-flop. As shown in this circuit, another decoder is inserted in the receiver domain of the circuit. The outputs of this decoder determines the number of delay buffers inserted in the clock path of the faulty flip-flop. As shown in Fig. 10, our error recovery mechanism is scalable.

Generally, if the receiver domain includes m flip-flops out of which n flip-flops reside in the clock boundary, the proposed error recovery circuitry includes $4n$ buffers, $5n$ tri-state buffers, n inverters, and $4n$ 2-input AND gates, one shift register (including $\lceil \log_2(n+1) \rceil$ registers), one decoder (with $\lceil \log_2(n+1) \rceil$ inputs), and one k-input decoder, where $k = \lceil \log_2(N) \rceil$, and N is the maximum number of delay buffers added to the clock path of the faulty flip-flop. In addition, one tri-state buffer is inserted in the sender domain and another one in the receiver domain to keep the relation of Clk_1 and Clk_2 in the error recoverable circuit equal to the relation of these values in the original circuit.

The proposed error recovery scheme requires that all flip-flops residing in the receiver side of clock boundary be equipped with the recovery circuitry during logic synthesis.

For each circuit under test, full-scan insertion is first performed. Next, we extract all connected pairs of flip-flops residing at clock boundaries. For each extracted receiver flip-flop, we insert the discussed error recovery circuitry in its clock path. As discussed, above one tri-state buffer is also inserted in the sender domain and another one in the receiver domain to preserve the relationship between the sender and receiver clock signals in the error-recoverable circuit. However, the proposed method requires extra input pins. For example, the circuit shown in Fig. 10 is an error-recoverable design of the circuit, shown in Fig. 8, with four extra primary input pins compared to the original circuit.

In general, to equip a multiclock circuit with the proposed CDC error recovery scheme, k+2 input pins need to be added to the original circuit, i.e., two extra input pins needed to feed *Clock* and *Data-in* inputs of the shift register feeding Decoder_1 with the specification of the faulty flip-flop, and $k=\lceil \log_2(N) \rceil$ input pins required to feed Decoder_2, with the specification of the number of buffers (N) added to the clock path of the faulty flip-flop.

To avoid inserting extra input pins in the error recoverable model of a multiclock circuit, the following steps are necessary. Assume that the CUT includes m flip-flops in the clock boundary, out of which n flip-flops reside in the receiver domain.

1) *Step 1*: Insert all flip-flops of the original circuit in scan-chain *Chain*1.
2) *Step 2*: Add $\lceil \log_2(n+1) \rceil + \lceil \log_2(N) \rceil$ registers to the CUT and insert these registers in scan-chain *Chain*2.
3) *Step 3*: Merge *Chain*1 and *Chain*2, i.e., connect the scan-out output of *Chain*2 to the scan-in input of *Chain*1. In addition, connect the functional/test signal of these two scan-chains to each other. In addition, connect the clock signals of these two scan chain to the receiver clock signal in the functional mode.
4) *Step 4*: Connect the Q output of each of the registers included in *Chain*2 to its D input, i.e., each register in *Chain*2 keeps its value in the functional mode.
5) *Step 5*: Extract all connected pairs of flip-flops residing at clock boundaries. Then, extract the receiver flip-flop of each CDC pair and put it in the recovery list $R-List$.
6) *Step 6*: Insert the recovery circuitry in the clock path of each flip-flop included in $R-List$. As discussed above, the recovery circuitry includes a set of buffers and tri-state buffers. There is a tradeoff between the number of buffers and tri-state buffers inserted for error recovery purpose and the recovery percentage. Obviously, the delay imposed to the clock path of the faulty flip-flop should not exceed the setup time of that flip-flop.
7) *Step 7*: Insert two decoders in the receiver domain, with $\lceil \log_2(n+1) \rceil$ and $\lceil \log_2 N \rceil$ inputs, respectively. Decoder_1 is fed with the first $\lceil \log_2(n+1) \rceil$ registers included in *Chain*1 and Decoder_2 is fed with the remaining $\lceil \log_2 N \rceil$ registers of *Chain*1. The outputs of Decoder_1 denote the ID of the faulty flip-flop (the value of zero is reserved for the case where no CDC fault is reported). In addition, the outputs of Decoder_2 indicate the number of buffers inserted in the clock path of the faulty flip-flop.
8) *Step 8*: Insert one common tri-state buffer in the clock path of all the flip-flops in the sender domain and another tri-state-buffer in the clock path of all flip-flops in the receiver domain other than the one included in $R-List$. As discussed above, these two tri-ate buffers are inserted to preserve the timing relationship between the sender and receiver clock signals.

To validate the proposed error-recovery method, we applied it to the circuit, shown in Fig. 1, and repeated the Monte Carlo simulation experiments. As shown in Table I, in more than

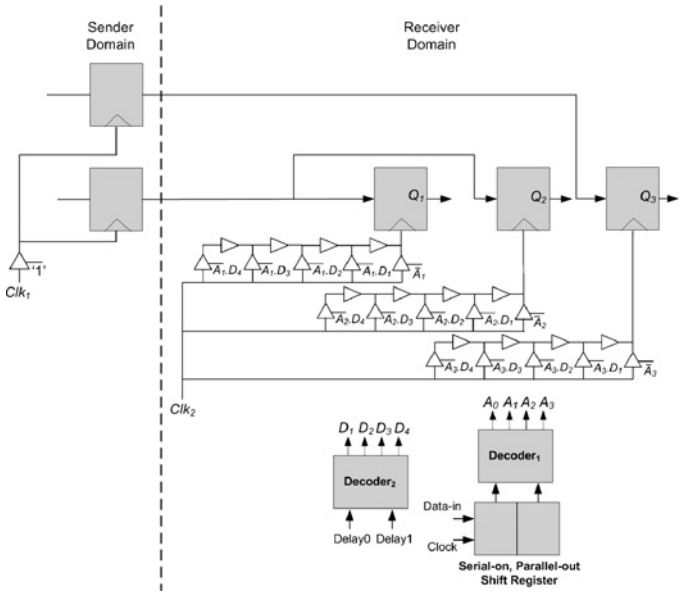

Fig. 10. Error-recoverable model of the circuit shown in Fig. 8 with up to four delay buffers in the clock path of the faulty flip-flop.

50% of the experiments for the original circuit, variation of the parameters of flip-flop DFF_3 result in a setup time violation at the receiver flip-flop. However, by inserting only one buffer in the clock path of flip-flop DFF_3, on an average more than 99.99% of the CDC errors can be recovered. All other CDC errors can be recovered if two buffers are employed in the clock path of DFF_3. In practice, by inserting one buffer in the path of Clk2, the number of runs with setup time violation is reduced to 8 out of 10 000 experiments.

As discussed above, the proposed recovery scheme is based on the use of tunable buffers [23], [24]. In this paper, we have extended the use of such buffers by designing a general low area-overhead circuitry to recover from S-CDC errors in multiple-clock domain circuits. As discussed, by using this circuitry, almost all S-CDC errors can be corrected.

2) *Applying the CDC Error-Recovery Method:* To perform error recovery, the following steps are taken. First, the fault-diagnosis scheme discussed in Section VI-A is applied to the recoverable CUT and the list of suspect locations (*S-List*) along with the test vectors detecting them are extracted from the CDC-fault dictionary. Then, the suspect locations are considered, one after the other, to determine the faulty location and recover from the CDC error. For each suspect location, different configurations of the circuit (insertion of 1 to N buffers in the clock path of the faulty flip-flop) are investigated. Finally, the configuration for which recovery from CDC error is possible, is selected. Upon selecting the configuration that allows recovery from CDC error, the ID of the faulty flip-flop as well as number of the buffers needed to be inserted in the clock path of the faulty flip-flop ($Decode_2$) are stored. These values are used for reconfiguring the circuit during normal operation.

VII. Experimental Results and Analysis

In this section, we first provide details of the simulation setup used to evaluate the effectiveness of the proposed schemes. Then, we present results on a number of IWLS'05 benchmarks, and discuss our observations.

A. Experimental Setup

To evaluate the effectiveness of the proposed fault detection, diagnosis, and recovery schemes, we use five different SIWLS'05 benchmarks that contain multiple clock domains. They are the WISHBONE AC 97 Controller (ac97_ctrl), the WISHBONE Memory Controller (mem_ctrl), the USB function core (usb_funct), the Ethernet IP core (ethernet), and the WISHBONE rev.B2 compliant Enhanced VGA/LCD Controller (vga_lcd)[27].

Software to perform scan insertion, CDC-path extraction, replication, selection of the final test patterns, generating fault dictionary, inserting error recovery circuit, and evaluating the results were all implemented using Python. A commercial ATPG tool was used for test generation and fault simulation. As indicated in our results, the ATPG tool reported a number of S-CDC faults to be untestable (or redundant). A commercial synthesis tool was used for synthesis and evaluating the effectiveness of the proposed recovery scheme.

To generate a test-pattern set that detects TDFs as well as S-CDC faults, top-off ATPG was performed after applying CoTC to meet the fault coverage requirement for TDFs. The final pattern set for our fault detection procedure, therefore, includes the CoTC-generated patterns and the top-off ATPG patterns. Note that top-off ATPG patterns do not detect any CDC faults beyond the CDC faults detected by CoTC.

All experiments were performed on a dual-processor Xeon quad-core Intel server running at 2.53 GHz with 64 GB of memory. The CPU time for CoTC was estimated by aggregating the times needed for the different steps. For the test cases in this paper, the test generation time per fault ranged from a few seconds to 3 min.

B. Experimental Results

In this section, the results of applying the proposed fault detection, diagnosis, and recovery methods to IWLS'05 benchmarks are presented and their significance are highlighted. The results are divided into four sets; the first set deals with the gate-level specification of each benchmark used in this study. The second set discusses the effectiveness of CoTC in detecting CDC faults. The third set evaluates the proposed fault diagnosis method. Finally, the fourth set evaluates the effectiveness of our error recovery scheme.

1) *Benchmark Statistics:* Details of the IWLS'05 benchmark circuits used in this paper are shown in Table III. The benchmarks represent a wide range of application areas, including memory controllers and IP cores. The ethernet benchmark has three clock domains, and all other benchmarks have two clock domains each. Note that in our experiments, we only considered slow-to-rise S-CDC faults. We expect to get similar results for slow-to-fall S-CDC faults without any change in methodology.

2) *Fault Detection Results:* This section highlights the effectiveness of CoTC in detecting CDC faults.

The first set of results compares the number of S-CDC faults detected by CoTC with the number of S-CDC faults detected by the baseline LoC/TDF method. For each benchmark circuit, we first extracted all CDC paths of the circuit, and then for

TABLE III
BENCHMARKS STATISTICS

Benchmark	# Clock domains	# All flip-flops	# Pairs of boundary flip-flops	# Gates
ac97_ctrl	2	2,199	902	28,083
mem_ctrl	2	1,083	3,354	22,015
usb_funct	2	1,746	1,592	25,531
ethernet	3	10,544	4,862	153,948
vga_lcd	2	17,079	3,187	252,302

each pair of the CDC flip-flops, we generated test patterns by applying CoTC to the time-expanded model of the circuit under test. The third column of Table IV shows the number of testable slow-to-rise S-CDC faults in each benchmark circuit. The fourth column of this table shows the number of slow-to-rise S-CDC faults detected by CoTC for each benchmark circuit.

To evaluate the number of slow-to-rise S-CDC faults detected by the baseline LoC/TDF method, we used a commercial ATPG tool to generate test patterns detecting all slow-to-rise TDFs for that benchmark. Then, the subset of the generated patterns that satisfied the constraints of CoTC scheme was extracted, and the number of slow-to-rise S-CDC faults detected by these vectors were reported (fifth column of Table IV). For each benchmark circuit, the sixth and the seventh columns of Table IV show the percentage of the testable S-CDC faults detected by CoTC and the baseline LoC/TDF method, respectively.

For the benchmark circuits considered in this paper, on average, the test patterns generated by CoTC can detect 88% of detectable S-CDC faults. We expect the fault coverage to be even higher since many faults that are aborted by the ATPG tool are most likely to be untestable. However, only 24% of the S-CDC faults can be detected using the baseline LoC/TDF method.

We next compare the number of slow-to-rise TDFs detected by LoC/TDF to the corresponding number for CoTC with top-off ATPG. The results are shown in Table V. The number of slow-to-rise TDFs detected by the traditional LoC method, is nearly equal to the number of transition faults detected by CoTC and top-off ATPG. Therefore, the proposed method provides the same coverage for TDFs as the baseline LoC/TDF method, but with a significantly higher coverage of CDC faults.

The next set of results compares the number of test patterns generated by LoC/TDF to the number of test patterns generated by CoTC with top-off ATPG. As shown in Table VI, on an average, for each circuit, the number of test patterns generated by CoTC with top-off ATPG is only 25% more than the patterns generated by using baseline LoC/TDF method. Therefore, higher test quality is attained with only a slight increase in test pattern count. As shown in Table VI, for the ethernet benchmark, the number of test patterns generated by CoTC with top-off ATPG is even less than the patterns generated while using baseline LoC/TDF method.

3) *Fault Diagnosis Results:* This section evaluates the proposed fault diagnosis method.

The next set of results evaluates the fault diagnosis rate of each benchmark while injecting slow-to-rise S-CDC faults. To

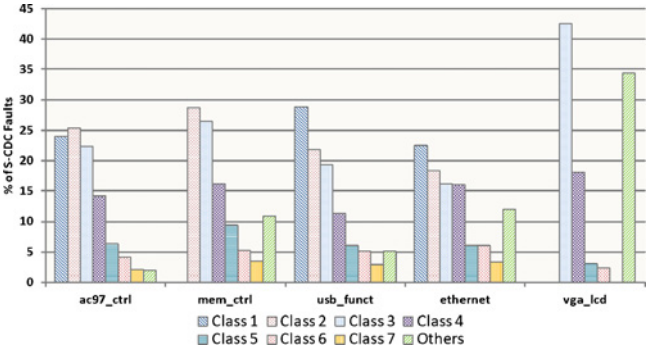

Fig. 11. Percentage of S-CDC faults classifying in each class of faults.

generate a CDC-fault dictionary for each benchmark circuit, we first apply CoTC to the circuit-under-test and extract the set of patterns detecting each S-CDC fault. Then, applying the steps discussed in Section VI-A, we generate a CDC-fault dictionary for that benchmark.

To evaluate the effectiveness of the proposed diagnosis method, we used the CDC-fault dictionary generated for each benchmark and simulated the CUT using all test patterns included in that fault dictionary in the presence of each S-CDC fault. Then, for each S-CDC fault f, the list of all S-CDC faults that can be distinguished from the fault f using the generated CDC-fault dictionary was extracted. The CDC faults were placed in different classes based on the number of faults from which they are distinguishable using the generated CDC-fault dictionary. If fault f is distinguishable from all other CDC faults, it is placed in Class 1; otherwise, it is placed in Class i, where i is the number of faults from which fault f cannot be distinguished using the generated fault dictionary.

Fig. 11 shows distribution of all CDC faults to different distinguishable classes for each benchmark circuit. As shown in this figure, on average for each benchmark circuit, 15% of detectable CDC faults are categorized as Class 1 fault, i.e., are fully diagnosable (23%, 28%, and 22% of faults in ac97_ctrl, usb_funct, and ethernet benchmarks are Class 1 faults, respectively. No faults is categorized as Class 1 fault in mem_ctrl and vga_lcd benchmarks). In addition, on average, in 72% of the cases, a list of two–seven suspicious locations are reported as the fault location. Note that in the vga_lcd benchmark, we consider a random sample of 175 faults because the large number of faults in this circuit makes exhaustive enumeration impractical. (Sampling has been done for the results shown in Fig. 11 and Table VII)

As discussed in Section VI-A, using the proposed fault diagnosis method, all S-CDC faults should be distinguishable from each other and categorized in Class 1. However, due to the limitation of the commercial ATPG tools used, we cannot generate all the test patterns detecting each CDC fault. As discussed in Section VI-A, we limited the number of test patterns generated by CoTC to detect each CDC fault to 255. Accordingly, two faults F_i and F_j that are reported as being distinguishable with test pattern T when the fault dictionary is generated, may not be distinguishable by that test pattern, and therefore may be categorized in a fault class other than Class 1 when the generated CDC-fault dictionary is used for fault diagnosis.

TABLE IV
COMPARING COTC AND TRADITIONAL LOC SCHEMES IN TERMS OF S-CDC FAULT DETECTION

Benchmark	S-CDC faults	# Testable S-CDC faults	# Detected by CoTC	# Detected by LoC/TDF	% Detected by CoTC	% Detected by LoC/TDF
ac97_ctrl	902	897	897	121	100	13
mem_ctrl	3,354	2,613	1,631	167	62	6
usb_funct	1,592	1,116	1,060	193	95	17
ethernet	4,862	643	529	391	82	61
vga_lcd	3,187	3,085	3,085	678	100	22

TABLE V
DETECTED SLOW-TO-RISE FAULTS

Benchmark	# Slow-to-rise faults	# Detected by LoC/TDF	# Detected by CoTC + top-off ATPG	% Detected by LoC/TDF	% Detected by CoTC + top-off ATPG
ac97_ctrl	40,916	37,154	37,140	90.80	90.77
mem_ctrl	38,086	17,266	17,482	45.33	45.90
usb_funct	40,108	34,718	34,850	86.56	86.89
ethernet	160,454	152,098	152,090	94.79	94.79
vga_lcd	382,927	317,092	317,074	82.81	82.80

TABLE VI
COMPARISON OF NUMBER OF TEST PATTERNS

Benchmark	LoC/TDF	CoTC	Top-off ATPG	CoTC + top-off ATPG	% increase
ac97_ctrl	1,591	412	1,468	1,880	18
mem_ctrl	1,094	846	979	1,825	66
usb_funct	2,414	576	2,107	2,683	11
ethernet	10,095	291	9,715	10,006	−1
vga_lcd	11,335	3,083	11,549	14,632	29

TABLE VII
DIAGNOSTIC EXPECTATION OF CDC-FAULT DICTIONARY

Benchmark	Diagnostic Expectation
ac97_ctrl	2.9
mem_ctrl	4.7
usb_funct	2.8
ethernet	4.0
vga_lcd	5.9

Diagnostic expectation is another metric used to evaluate the diagnostic capability of a test set or test sequence [28]. Diagnostic expectation is the weighted average size of the indistinguishability classes obtained using a test-pattern set [29]. The smaller the diagnostic expectation, the higher the precision of the diagnosis. Table VII presents the diagnostic expectation of the CDC-fault dictionaries generated for each benchmark circuit.

We also evaluate the size of CDC-fault dictionary generated for each benchmark circuit in terms of its pattern count. The second column of Table VIII represents the test-pattern count of the CDC-fault dictionary generated for each benchmark circuit. We compare the number of test patterns included in the fault dictionary of each CUT with the number of test pattern generated by CoTC method to detect all S-CDC faults (third column of Table VI). Results indicate that on average, for each circuit, the number of test patterns included in the fault dictionary is only 31% more than the patterns generated using CoTC.

TABLE VIII
NUMBER OF TEST PATTERNS INCLUDED IN THE CDC-FAULT DICTIONARY

Benchmark	Pattern count	% increase vs. CoTC
ac97_ctrl	549	33
mem_ctrl	1,429	69
usb_funct	732	27
ethernet	376	29
vga_lcd	3,084	0

4) *Error Recovery Results:* We next evaluate the proposed error recovery scheme.

To evaluate the effectiveness of the error recovery method proposed in Section VI-B, we synthesized each benchmark using a commercial logic synthesis tool and targeting a Nangate 45-nm library. For each benchmark circuit, we considered the minimum clock frequencies under which the circuit can operate properly.

By considering the propagation delay of the logic residing between the output signal of the boundary flip-flops in the receiver domain and the input of the flip-flops fed by that signal, we evaluated the number of delay buffers that can be inserted in the clock path of the boundary flip-flops to mask S-CDC faults. Table IX shows our results. As indicated in this table, at least seven delay buffers can be inserted in the clock path of boundary flip-flops. Comparing the delay of a buffer element in the target library with the setup time of the flip-flops in the same library, we find that the number of permitted delay buffers (seven) always exceeds the number of delay buffers required to mask CDC faults. Therefore, 100% of detectable S-CDC faults can be masked in each benchmark circuit by applying the proposed error recovery scheme.

The final set of results focuses on the cost-effectiveness of the proposed error recovery method. We synthesized each benchmark before and after applying our error-recovery scheme using a commercial logic synthesis tool and targeting a Nangate 45-nm library. Results are shown in Table X. The third column of this table shows the total area of each benchmark, when we insert at most one buffer in the clock path of the boundary flip-flops. The area overhead numbers shown

TABLE IX
NUMBER OF DELAY BUFFERS THAT CAN BE INSERTED IN THE CLOCK PATH OF BOUNDARY FLIP-FLOPS TO RECOVER FROM CDC ERRORS

Benchmark	# of delay buffers
ac97_ctrl	7
mem_ctrl	50
usb_funct	16
ethernet	18
vga_lcd	38

TABLE X
AREA OVERHEAD INCURRED BY PROPOSED ERROR RECOVERY SCHEME

Benchmark	Area of original circuit (in μm^2)	Area of circuit with error recovery (in μm^2)	% increase
ac97_ctrl	35,002.4	10,885.7	31.1
mem_ctrl	23,258.7	6,419.4	27.6
usb_funct	32,151.7	3,311.6	10.3
ethernet	188,493.7	15,644.9	8.3
vga_lcd	307,111.6	1,842.6	0.6

in this table are calculated using data reported by the logic synthesis tool. As shown in this table, on an average, the area overhead of the error-recovery scheme is 15.5%. The area overhead is considerably less for the two largest benchmark circuits.

In Table X, the area overhead of applying our error-recovery method to *ac97_ctrl* and *mem_ctrl* benchmarks is more than the other benchmarks. In fact in these two circuits, nearly 30% of the flip-flops are in the receiver boundary. In other benchmarks, less than 10% of the flip-flops reside in the receiver boundary. As mentioned in Section VI-B, for each circuit with n boundary flip-flops residing in the receiver domain, the error recovery circuitry includes n buffers, $2n+2$ tri-state buffers, n inverters, one shift register (including $\lceil \log_2(n+1) \rceil$ registers) and one decoder (with $\lceil \log_2(n+1) \rceil$ inputs). Therefore, since the percentage of boundary flip-flops that reside in the receiver domain for *ac97_ctrl* and *mem_ctrl* benchmarks are much more than that for the other benchmarks, the area overhead of applying our error recovery method to these two benchmark circuits is more than the others.

VIII. CONCLUSION

Even when robust design methods based on synchronizers and design verification techniques were used, process variations could introduce subtle timing problems that affect data transfer across clock-domain boundaries for fabricated chips. Accordingly, modeling the incorrect behavior of multiclock circuits in the presence of CDC faults, detecting and locating such faults, and recovery from CDC failures were necessary. We presented a test generation method for detecting CDC faults. Fault diagnosis was performed by employing a CDC fault dictionary. While a CDC fault was located, its impact was masked using post-silicon clock-path tuning. We applied our CDC fault detection, diagnosis, and recovery schemes to the IWLS'05 benchmark circuits with multiple clock domains. The results highlighted the effectiveness of the proposed methods in the recovery of multiclock circuits from CDC failures.

REFERENCES

[1] N. Karimi, K. Chakrabarty, P. Gupta, and S. Patil, "Test generation for clock-domain crossing faults in integrated circuits," in *Proc. Des. Automation Test Eur. Conf.*, 2012, pp. 406–411.

[2] Y. Feng, Z. Zhou, D. Tong, and X. Cheng, "Clock domain crossing fault model and coverage metric for validation of SoC design," in *Proc. Des. Automation Test Eur. Conf.*, 2007, pp. 1–6.

[3] R. Ginosar, "Fourteen ways to fool your synchronizer," *Asynchronous Circuits Syst.*, vol. 1, pp. 89–96, 2003.

[4] S. Sarwary and S. Verma, "Critical clock-domain-crossing bugs," *Electron. Des. Strategy News*, vol. 53, no. 7, pp. 55–60, Apr. 2008.

[5] C. Kwok, V. Gupta, and T. Ly, "Using assertion-based verification to verify clock domain crossing signals," in *Proc. Des. Verification Conf.*, 2003, pp. 654–659.

[6] T. Kapschitz and R. Ginosar, "Formal verification of synchronizers," in *Correct Hardware Design and Verification Methods*, vol. 3725, D. Borrione and W. Paul, Eds. Berlin, Germany: Springer, 2005, pp. 359–362.

[7] N. Karimi, Z. Kong, K. Chakrabarty, P. Gupta, and S. Patil, "Testing of clock-domain crossing faults in multi-core system-on-chip," in *Proc. Asian Test Symp.*, 2011, pp. 7–14.

[8] R. Ginosar, "Metastability and synchronizers: A tutorial," *IEEE Des. Test Comp.*, vol. 28, no. 5, pp. 23–35, Sep. 2011.

[9] H.-K. Kim, L.-T. Wang, Y.-L. Wu,·and W.-B. Jone, "Testing of synchronizers in asynchronous FIFO," *J. Electron. Testing Theory Appl.*, vol. 29, no. 1, pp. 49–72, 2013.

[10] J. M. Bassam, "Zero setup time flip-flop," U.S. Patent 5867049, Feb. 1999.

[11] K.-K. Kwok, B. Li, T. A. Ly, and R. R. Sabbagh, "Formal verification of clock domain crossings," U.S. Patent 20100199244, Aug. 5, 2010.

[12] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits, A Design Perspective*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2003.

[13] B. Abramov. *Clock Domain Crossing* [Online]. Available: http://www.abramovbenjamin.net/malas/l9.pdf

[14] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm early design exploration," *IEEE Trans. Electron Devices*, vol. 53, no. 11, pp. 2816–2823, Nov. 2006.

[15] A. Murakami, S. Kajihara, T. Sasao, I. Pomeranz, and S. M. Reddy, "Selection of potentially testable path delay faults for test generation," in *Proc. Int. Test Conf.*, 2000, pp. 376–384.

[16] M. Litterick, "Pragmatic simulation-based verification of clock domain crossing signals and jitter using SystemVerilog assertions," in *Proc. Des. Verification Conf.*, 2006, pp. 1–6.

[17] N. Devtaprasanna, A. Gunda, P. Krishnamurthy, S. M. Reddy, and I. Pomeranz, "A novel method of improving transition delay fault coverage using multiple scan enable signals," in *Proc. Int. Conf. Comp. Des.*, 2005, pp. 471–474.

[18] Y. Benabboud, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, and O. Riewer, "Delay fault diagnosis in sequential circuits," in *Proc. Asian Test Symp.*, 2009, pp. 355–360.

[19] J.-W. Chen, Y.-Y. Chen, and J.-J. Liou, "Handling pattern-dependent delay faults in diagnosis," in *Proc. VLSI Test Symp.*, 2007, pp. 151–157.

[20] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing & Testable Design*. New York, NY, USA: Wiley-IEEE Press, 1990.

[21] M. A. Shukoor and V. D. Agrawal, "Diagnostic test set minimization and full-response fault dictionary," *J. Electron. Testing Theory Appl.*, vol. 28, no. 2, pp. 177–187, Apr. 2012.

[22] N. K. Jha and S. Gupta, *Testing of Digital Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[23] E. Takahashi, Y. Kasai, M. Murakawa, and T. Higuchi, "A post-silicon clock timing adjustment using genetic algorithms," in *Proc. Symp. VLSI Circuits*, 2003, pp. 13–16.

[24] K. Nagaraj and S. Kundu, "An automatic post silicon clock tuning system for improving system performance based on tester measurements," in *Proc. Int. Test Conf.*, 2008, pp. 1–8.

[25] Y. Elboim, A. Kolodny, and R. Ginosar, "A clock-tuning circuit for system-on-chip," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 11, no. 4, pp. 616–626, Aug. 2003.

[26] T. Saeki, Y. Nakaoka, M. Fujita, A. Tanaka, K. Nagata, K. Sakakibara, T. Matano, Y. Hoshino, K. Miyano, S. Isa, E. Kakehashi, J. M. Drynan, M. Komuro, T. Fukase, H. Iwasaki, J. Sekine', M. Igeta', N. Nakanishi', T. Itani, K. Yoshida, H. Yoshino, S. Hashimoto, T. Yoshii, M. Ichinose, T. Imura, M. Uziie, K. Koyama, Y. Fukuzo, and T. Okuda, "A 2.5 ns clock access 250 MHz 256 MB SDRAM with a synchronous mirror delay," in *Proc. Int. Solid State Circuits Conf.*, vol. 476. 1996, pp. 374–375.

[27] C. Albrecht, "IWLS 2005 benchmarks," in *Proc. Int. Workshop Logic Synthesis*, 2005.[Online]. Available: http://iwls.org/iwls2005/benchmarks.html

[28] Y. Shao, R. Guo, I. Pomeranz, and S. M. Reddy, "The effects of test compaction on fault diagnosis," in *Proc. Int. Test Conf.*, 1999, pp. 1083–1089.

[29] P. G. Ryan, W. K. Fuchs, and I. Pomeranz, "Fault dictionary compression and equivalence class computation for sequential circuits," in *Proc. Int. Conf. CAD*, 2003, pp. 508–511.

Naghmeh Karimi (M'05) received the B.S., M.S., and Ph.D. degrees in computer engineering from the University of Tehran, Tehran, Iran, in 1997, 2002, and 2010, respectively. Her Master's thesis was on testability enhancement at the register transfer level and her Ph.D. thesis was on concurrent error testing and reliability enhancement.

Between 2007 and 2009, she was a Visiting Researcher at Yale University, New Haven, CT, USA. She was a Post-Doctoral Researcher at Duke University, Durham, NC, USA, for one year. She is currently a Visiting Assistant Professor at the Polytechnic Institute, New York University, Brooklyn, NY, USA. Her current research interests include design-for-testability, concurrent testing, fault tolerance, reliability enhancement, and hardware security.

Krishnendu Chakrabarty (F'08) received the B.Tech. degree from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, MI, USA, in 1992 and 1995, respectively.

He is currently a Professor of electrical and computer engineering at Duke University, Durham, NC, USA. He is also a Chair Professor at Tsinghua University, Beijing, China, a Visiting Chair Professor at National Cheng Kung University, Tainan, Taiwan, and a Guest Professor at the University of Bremen, Bremen, Germany. He holds two U.S. patents and has several pending patents. His current research interests include testing and design-for-testability of integrated circuits, digital microfluidics, biochips, and cyberphysical systems, optimization of digital print, and enterprise systems. He has authored 12 books on these topics (with two more books in press), and has published over 440 papers in journals and refereed conference proceedings.

Dr. Chakrabarty is a recipient of the National Science Foundation Early Faculty (CAREER) Award, the Office of Naval Research Young Investigator Award, the Humboldt Research Fellowship from the Alexander von Humboldt Foundation, Germany, and several Best Paper Awards at IEEE conferences. He has given over 190 invited, keynote, and plenary talks. He has also presented 30 tutorials at major international conferences. He is a a Golden Core Member of the IEEE Computer Society, and a Distinguished Engineer of ACM. He was a 2009 Invitational Fellow of the Japan Society for the Promotion of Science. He was a recipient of the 2008 Duke University Graduate School Dean's Award for excellence in mentoring, and the 2010 Capers and Marion McDonald Award for Excellence in Mentoring and Advising, Pratt School of Engineering, Duke University. He served as a Distinguished Visitor of the IEEE Computer Society from 2005 to 2007 and 2010 to 2012, and as a Distinguished Lecturer of the IEEE Circuits and Systems Society from 2006 to 2007. He currently serves as an ACM Distinguished Speaker and a Distinguished Lecturer of the IEEE Circuits and Systems Society from 2012 to 2013. He served as the Editor-in-Chief of the IEEE DESIGN and TEST of COMPUTERS from 2010 to 2012. Currently, he serves as the Editor-in-Chief of the *ACM Journal on Emerging Technologies in Computing Systems*. He is also an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN of INTEGRATED CIRCUITS and SYSTEMS, the IEEE TRANSACTIONS on COMPUTERS, IEEE TRANSACTIONS on CIRCUITS and SYSTEMS II, and the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS and SYSTEMS. He serves as an Editor of the *Journal of Electronic Testing: Theory and Applications*. In the recent past, he has served as an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS and IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN of INTEGRATED CIRCUITS and SYSTEMS, IEEE TRANSACTIONS on CIRCUITS and SYSTEMS I.